Mobile Development

> This document contains practice task for Mobile Development course, year 2024. This course consists of four exciting practice tasks that will challenge your mobile development skills. Get ready to dive into the world of mobile app development, we're all about building high-quality, production-ready apps, so make sure to showcase your clean and well-designed code.

Common Info

1. **Technologies. For this course you are allowed to use:**
    a. Kotlin/Java – for Android Platform
    b. Swift/Objective-C – for IOS Platform
    c. No cross-platform stuff as Flutter, React Native, Maui is allowed.
        i. This one could be discuss personally if student works in a mobile development.
2. **General Acceptance Criteria.**
    a. Git Repo.
    b. Production Build.
    c. Clean architecture.
    d. Functional Design.
    e. Task related theoretical knowledge.

After completing the following four steps, you will have a functional application that can be presented as your final project for this course. It would be great if you have your own ideas. Let's discuss them, and together we will find a suitable way to bring them to life. =)

General overview

The main topic for your project is gonna be simple as F. One and only Calculator, supports base functions.

Step 1: Common Functionality

Welcome to the exciting world of Mobile Development! In this journey, we will explore the ins and outs of this fascinating field. During work#1, you will have the opportunity to delve into platform-specific aspects of mobile development.

For this Step, your main goal is going to be to implement the basic functionality for the calculator, while adhering to the latest mobile development standards. In order to achieve this, you will need to carefully analyze the requirements and design an intuitive user interface that allows users to perform various calculations effortlessly.

Additionally, it is ***crucial to ensure that the calculator is compatible with different screen sizes and resolutions***, providing a seamless experience across a wide range of mobile devices. As part of the implementation process, you should also consider incorporating error handling mechanisms to gracefully handle any unexpected user inputs or system errors.

Acceptance criteria

1. Base calculator logic is implemented.
2. Usage of base kits (IOS/Android) is allowed. (It would be a plus if u decided to build your custom components).
3. Knowledge of the project structure.
4. Knowledge of the OS layers.
5. App Icon / Splash Screen is required.

Useful links

- https://developer.android.com/courses/android-basics-compose/course
- https://developer.android.com/develop/ui/views/layout/declaring-layout
- https://developer.apple.com/design/human-interface-guidelines/patterns
- https://docs.swift.org/swift-book/documentation/the-swift-programming-language/
- https://developer.android.com/topic/architecture
- https://medium.com/@wil.barriost/ios-clean-architecture-my-way-on-a-very-simple-feature-mvvm-repository-urlsession-swift-678cfe4301f0
- https://developer.apple.com/ios/planning/

Step 2: Platform API

For this work, your primary objective is to implement various functionalities that can be achieved by utilising the application programming interface (API) of the mobile operating system. This includes but is not limited to integrating different features, optimising performance, and enhancing user experience. By leveraging the capabilities of the mobile OS API, you can unlock a wide range of possibilities to enhance the functionality and overall performance of the application.

E.g. Interesting inputs, handling Touch Events, Widgets, Camera and so on.

Acceptance criteria

1. Usage of Platform API is added.
    a. You can choose your own feature.

Useful links

- https://developer.android.com/develop#core-areas
- https://developer.apple.com/app-extensions/

Step 3: Fetching

Hey there! Fetching is something that you should definitely become best friends with as you near the end of your 3rd year. Your 3rd project should focus on enabling communication with the outside world. To achieve this, you'll need to connect your app to a cloud platform. I suggest taking a look at Firebase Cloud Platform - it's a great option!

For this step, your main goal is to implement the fetching functionality in your calculator app. This involves connecting your app to a cloud platform, such as Firebase, and enabling communication with external data sources. You can use this functionality to retrieve data from the cloud, such as theme customization parameters, which should be saved in the cloud. Additionally, you should implement the push notification API to enhance the user experience and keep them updated with relevant information.

Make sure to save and load action histories from cloud storage. If you are planning to use neural networks for input recognition, it is advisable to transfer this functionality to a cloud function.

Acceptance criteria

1. Theme customisation feature (Theme parameters should be saved in the cloud)
    a. The progress bar color should be the same color as specified in the theme.
2. Implement the push notification API.
3. Load/saving action history feature.

Useful links

- Firebase Cloud Platform

Step 4: Pass Key Authorization

For this task, you will be implementing a Pass Key authorization feature to enhance the security of the mobile app. The Pass Key will serve as a password or PIN that the user needs to enter in order to access certain features or sensitive information within the app.

Acceptance Criteria

1. Initiallize Pass Key during app setup.
2. Handle scenarios where the entered Pass Key is incorrect or forgotten.
3. Implement logic to validate the entered Pass Key and grant access to authorized features upon successful validation.

4. Provide an option for the user to reset or change their Pass Key.
5. Pass Key is securely stored and protected against unauthorized access.
6. Use Biometric Authentication APIs.

Useful links

- Android KeyStore
- iOS Keychain Services
- Android Pass Key